兼顾随机性与时间效率的图数据构造算法

葛昌威，王桂平，黄春淦

（重庆交通大学信息科学与工程学院，重庆 404100）

摘要：图数据是典型的非欧数据，因其对结构化信息的天然内在编码在近年来受到许多关注。现实中的社交网络、化学分子、交通线路图和知识图谱等都可以很自然地表示为这种非结构化的数据。许多的研究基于图数据之上进行复杂的建模。然而，对于图数据的构造算法却鲜有研究。因此，本文针对常见的图数据模型，分别对树、有向无环图、有向带环图、二分图和仙人掌图提出了兼顾随机性与时间效率的构造算法，并给出了相关正确性与复杂度的证明。本文的算法均附有伪代码，相关实现代码已开源至 GitHub。

关键词：图数据；构造算法；二分图；仙人掌图

Graph construction algorithms considering randomness and temporal efficiency

Ge Changwei, Wang Guiping, Huang Chungan

(College of Information Science and Engineering，Chongqing jiaotong University，Chongqin，China，404100)

Abstract： Graph data, as typical non-European data, has attracted much attention in recent years because of its intrinsic encoding of naturally structured information. Social networks, chemical molecules, maps of traffic routes and knowledge maps can all be represented naturally as this kind of unstructured data. Many studies have done complex modelling based on graphs. However, there is little research on the graph construction algorithm. Therefore, in this paper, for common graph data models, the randomness and time efficiency of the tree, directed acyclic graph, directed ring graph, bipartite graph and cactus graph are respectively proposed, and the correlation correctness and complexity are proved. All the algorithms in this paper are accompanied by pseudo code, and relevant implementation codes have been collated to GitHub.

Keywords： graph data; graph Construction algorithm; Bipartite graph; Cactus graph

0 引言

近年来，深度学习依靠其强大的学习表征能力在欧式数据领域（如图像、视频、声音、文本信息等）取得了令人瞩目的成就，然而，现实中的许多信息本质是非欧式数据，例如，图数据。图数据是一种具有强大表征能力的数据类型，社交网络、交通网络、蛋白质分子网络等都可以很自然的表示为图数据。然而，由于图数据不满足平移变换的结构特殊性，深度学习技术并不能直接从欧式数据迁移到图数据这类非欧数据上来。因此，图深度学习技术应运而生。

越来越多重要的深度学习任务涉及图结构数据集，例如对社交网络中的社交推荐[1]和异常账户检测[2]和知识图谱的构建[3]等。解决这些任务的主要难点在于如何找到正确的方式来表达和利用图的底层结构信息。传统上，这是通过计算各种图数据上的统计数据（如度数和中心性）、使用图形内核或提取人工工程特征来实现的[4]。由此可见目前许多的对于图的研究集中在基于图数据模型之上进行复杂的建模，如图像检索[5]、数据挖掘[6]、信息安全[7]等领域。而对图数据本身模型的构造算法研究较少，并且缺乏构造各种图模型的算法[8-9]，并且这会带来图上任务始终处于次优的问题，有时甚至会导致训练模型无法收敛。

常见的图数据模型大体上可以分为树，有向无环图，有向带环图，二分图，仙人掌图等。目前的构造方法存在一些随机性不强，无法构造边界数据或构造边界数据时复杂度会退化的问题。因此本文将按边权的构造，一般图的构造，特殊图的构造依次进行介绍，并给出兼顾随机性与时间效率的改进算法与 C++实现。本文所提出的所有算法的空间复杂度都是 O(n)级的。受限于篇幅，本文所提出的更多图的构造代码请参见 https://github.com/gcwAndHisFriends。

1 边权的构造

1.1 构造算法及证明

构造图的边权等价于构造一个数列，该数列由数列长度M，值域下界L，值域上界R，数列中数值是否可重复$Unique$，四个限制条件生成。若$Unique$为$true$，使用随机数生成器构造。若$Unique$为$false$，考虑下述两种算法。值得注意的是，本节的主要工作是区分在不同数据范围下，如何选择这两种算法，使得期望时间复杂度更低，并推导出了算法 1.3 中的判定式。

算法1.1：使用随机数生成器产生数据，通过重复值检查后加入到集合。

算法1.2：将值域中所有整数数值存入数组，随机打乱数组后取其中M项。

算法 1.1 适合构造稀疏数据，算法 1.2 适合构造稠密数据。下文提出了根据不同数据范围选择上述两种算法的界定算法及证明。

算法 1.3：设$N = R - L + 1$，当$ln\ (N+1) \le 1 + ln\big((N-M)+1\big)$时，采用算法 1.1，反之采用算法1.2。现证明如下：

如果已经得到了X个不同的数字，下一次随机取数结果是一个未出现数字的概率为$\frac{N-X}{N}$。在值域中进行若干次随机取值，得到M个不同数字的期望次数(算法1.1期望时间复杂度)为$\sum_{i=1}^{M}\frac{N}{N-i+1}$，算法1.2时间复杂度为$O(N)$。将期望时间复杂度作为判断条件，得判别式$\sum_{i=1}^{M}\frac{1}{N-i+1} \le 1$。若为真采用算法1.1。反之采用算法1.2。

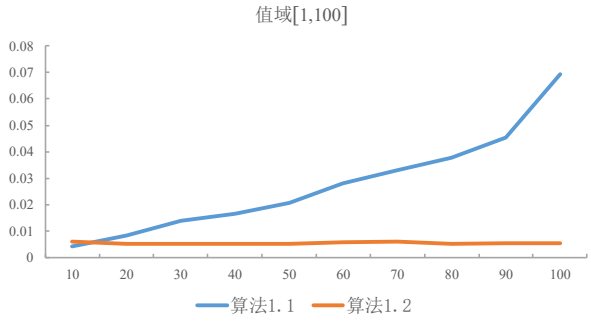
判别式计算时间复杂度为$O(M)$。考虑近似值优化：注意到当$M = N$，判别式左式为调和级数。调和级数有不等式[3]：$ln(N+1) \le \sum_{i=1}^{N}\frac{1}{N-i+1} \le (ln\ N) + 1$。将不等式带入到判别式左值，得$\sum_{i=1}^{M}\frac{1}{N-i+1} \approx ln\ (N+1) - ln\big((N-M)+1\big)$，将左式近似值带入到判别式，算法得证。

1.2 时空复杂度分析

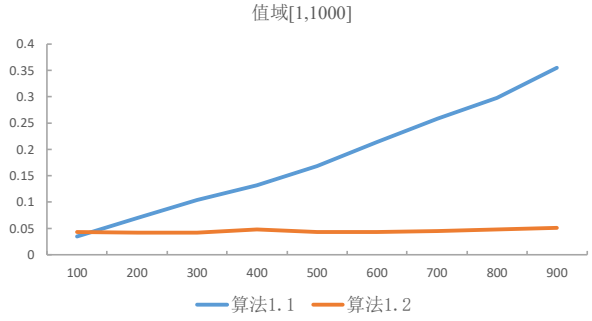
算法 1.1 空间开销在于重复值检查。如果使用哈希表进行检查，空间复杂度为$O(n)$，单次均摊查询复杂度为$O(1)$，总时间复杂度为$O\big(Mlog(N)\big)$。算法 1.2 空间开销在于存储值域$O(N)$，时间开销在于洗牌算法[10]$O(N)$。

1.3 实验结果分析

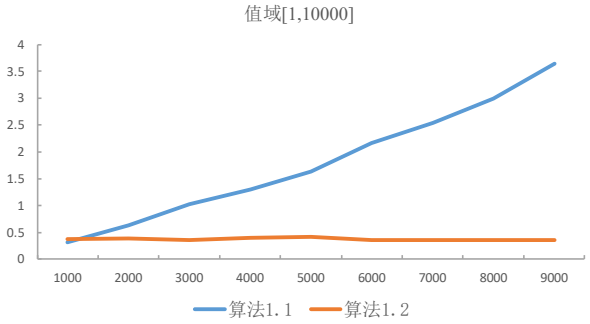
下图1展示了算法1.1和算法1.2在不同值域下，生成不同数量随机数的平均用时(单位:毫秒)。



(a)



(b)



(c)

图 1 算法 1.1 和算法 1.2 在不同值域下，生成不同数量随机数的平均用时(单位:毫秒)

可以看出在实际测试中，当$M > N/10$，算法1.1的耗时超过了算法1.2。实验结果展示的算法的实际运行耗时受编程语言、实现细节、CPU 性能等因素的影响。表1展示了算法1.2在不同值域范围下的运行耗时(单位：毫秒)。

表 1 算法1.2在不同值域范围下的运行耗时(单位：毫秒)

值域范围	100	1000	10000	100000	1000000	10000000
耗时(ms)	0.008	0.044	0.454	4.272	59.033	1302.48

构造边权算法伪代码如下算法 1 所示：

Algorithm 1 Construction of the Sequence

Input: array size m;Lower limit of value L ;Lower limit of value R;whether the values is unique
Output: array a

$a \leftarrow EmptyArray()$
$n \leftarrow R - L + 1$
if *unique is true* **then**
    if $(\ln(n+1) < 1 + \log((n-m)+1))$ **then**
        for $i \leftarrow 1$ *to* m **do**
            while u *exists in* a **do**
                $u \leftarrow rand(L, R)$
            end while
            append Integer u *to* a
        end for
    else
        for $i \leftarrow L$ *to* R **do**
            append i *to* a
        end for
        $a \leftarrow shuffle(a[1, m])$
    end if
else
    for $i \leftarrow L$ *to* R **do**
        $u \leftarrow rand(L, R)$
        while u *exists in* a **do**
            $u \leftarrow rand(L, R)$
        end while
        append Integer u *to* a
    end for
end if
return a

2 简单有向图的构造

2.1 构造简单有向图的算法及证明

本节将第一节中的算法 1.3 中给出的构造方法应用到有向图的构造中。非连通图由多个连通图组合而成，故此仅讨论连通图构造方法。假设要构造一个包含N个节点，M条边的图。按算法 2 构造一个基本边集$edge$保证图的连通性[11]。

Algorithm 2 Construction of the Tree

Input: Number of nodes of the tree n
Output: Pairs Arrays edge

for $i \leftarrow 2$ *to* n **do**
    $u \leftarrow rand(1, i-1)$
    $v \leftarrow i$
    $edge$ *append*$\{u, v\}$
end for
return $edge$

$deep_x$表示节点x在有根树中的深度。选择任意两点u，v，添加有序顶点对$\{u,v\}$至边集。当且仅当所有有序顶点对$\{u_i,v_i\}$，都有$deep_{u_i} < deep_{v_i}$，那么构造出的图为有向无环图。只要存在有一个有序点对$\{u_i,v_i\}$，满足$deep_{u_i} > deep_{v_i}$，那么构造出的图为有向带环图。有向带环图具体构造方法可参见算法 3。

Algorithm 3 Construction of the Directed Cycling Graph

Input: Number of nodes of the graph n;Number of edge of the graph m
Output: Pairs Arrays edge

```
for i ← 2 to n do
    u ← rand(1, i − 1)
    v ← i
    edge append {u, v}
end for
Upper limit ← n ∗ (n − 1)
if ln(Upper limit + 1) < 1 + log((Upper limit − m) + 1) then
    no ring ← true
    for i ← n to m do
        while u = v or edge exists {u, v} do
            u ← rand(1, n)
            v ← rand(1, n)
            if no ring = ture and u < v then
                swap(u, v)
            end if
        end while
        edge append {u, v}
        no ring ← false
    end for
else
    a append {j, i} where 1 ≤ i < j ≤ n and {j,i}
 not exists in a
    while size(a) < m do
        a append {i, j} where {i, j} not exists in a
    end while
end if
return a
```

简单图要求不能存在重边与自环，在构造边时需根据边密度进行构造。设$Upper\ \ limit$为图的最大边容量上限，有向无环图的$Upper\ \ limit$为$\frac{n*(n-1)}{2}$，有向带环图的$Upper\ \ limit$为$n*(n-1)$。由算法1.3，当$ln\ (Upper\ \ limit+1) \le 1+ln\big((Upper\ \ limit-M)+1\big)$，可以选择在拓扑序中随机选点，判重后连边的方法。反之，枚举出所有边，打乱后随机取其中一部分边的方法更合适。有向无环图具体构造方法可参算法 4。

Algorithm 4 Construction of the Directed Acyclic Graph

Input: Number of nodes of the graph n;Number of edge of the graph m
Output: Pairs Arrays edge

```
for i ← 2 to n do
    u ← rand(1, i − 1)
    v ← i
    edge append {u, v}
end for
Upper limit ← n ∗ (n − 1)/2
if ln(Upper limit + 1) < 1 + log((Upper limit − m) + 1) then
    for i ← n to m do
        while u = v or edge exists {u, v} do
            u ← rand(1, n)
            v ← rand(1, n)
            if u > v then
                swap(u, v)
            end if
        end while
    end for
else
    while size(a) < m do
        a append {i, j} where 1 ≤ i < j ≤ n and {i, j} not exists in a
    end while
end if
return a
```

2.2 时空复杂度分析

构造稠密图需要存储所有边，故空间开销为$O(N^2)$，由于需要对所有边进行洗牌，故时间开销为$O(N^2)$。构造稀疏图只需要存储M条边并对后续新增边进行判重，同样可以采用哈希表实现，空间复杂度为$O(M)$，时间复杂度为$O(MlogM)$。

3 特殊图的构造

3.1 二分图的构造

3.1.1 构造二分图的算法及其证明

节点由两个集合组成，且两个集合内部没有边的图被称为二分图。

算法3.1.1：将所有点随机分为两个集合，在两个集合之间随机选点连边。

算法3.1.1保证了高随机性，但构造出的二分图无法保证连通。下文给出了保证连通性的二分图构造算法并证明了正确性。

算法3.1.2：构造一棵节点数为N的树，称原树边集。取任意节点为根。设$deep_x$表示节点x的深度，f_x为节点x的父节点，$deep_x = deep_{f_x} + 1$。对于任意节点P，

$$P \in \begin{cases} Set_{prime}, & deep_p \mod 2 = 1 \\ Set_{composite}, & deep_p \mod 2 = 0 \end{cases}$$

随机取两节点u，v，若满足$u \in Set_{prime}$，$v \in Set_{composite}$，$\{u,v\} \notin edge_{tree}$，则添加$\{u,v\}$至新增边集。

引理3.1.1：如果一个图中不存在奇环，那么该图就是二分图，否则该图不是二分图[12]。

证明3.1.1：仅包含新增边$\{u,v\}$与部分原树边的环为偶环。

设两点分别为u，v，$distance(u,v)$为两点在原树上的距离，$LCA(u,v)$表示两点在原树上的最近公共祖先。因为树中两点之间路径唯一，并且$deep_u + deep_v = 1 \ (mod2)$，$distance(u,v) + 1 = (deep_u + deep_v - 2 \times deep_{LCA(u,v)}) + 1 = 0 \ (mod2)$，新增边$\{u,v\}$与部分原树边的环为偶环得证。

定义L是一个由K条新增边和部分原树边组成的环，L包含的第i条新增边为$\{x_i,y_i\}$。$loop_i$为新增边$\{x_i,y_i\}$与部分原树边构成环的边集。$loop_{i+j} = loop_i \oplus \mathrm{loop_j}$，$\oplus$表示两个边集的对称差运算。

证明3.1.2：$loop_i$与$loop_j$最多只有一段路径重叠。

采用反证法，如果存在两段路径重叠，由于$\{x_i,y_i\} \neq \{x_j,y_j\}$，所以原树上的两条路径$x_i \to y_i$，$x_j \to y_j$之间有两段重叠路径。设$u$，$v$分别是这两段重叠路径中的点，则$u$可以通过原树上两条不同路径到达$v$，与树的性质相悖，故假设不成立。三段以上重叠区域的证明同理。

证明3.1.3：$loop_{i+j}$表示一个由$\{x_i,y_i\}$，$\{x_j,y_j\}$与部分原树边构成的，具有唯一性的环(后续均不讨论$loop_{i+j}$不存在的情况)。

证明$loop_{i+j}$为环：由证明3.1.2可知，$loop_i$与$loop_j$最多只有一段连续路径重叠，设路径两端节点分别为u，v。$loop_i$与$loop_j$非重叠的边集所对应的点集除u，v外两两不同，将并集可以表示为仅有一对重复点对的顶点序列，并且该点集可以对应$loop_{i+j}$的边集，故$loop_{i+j}$为环。

证明$loop_{i+j}$具有唯一性：如果存在两个环，均仅包含$\{x_i, y_i\}$与$\{x_j, y_j\}$与部分原树边，并且至少存在一条边不同。说明点集$\{x_i, x_j, y_i, y_j\}$中至少存在一个点对，在原树边集中的路径不唯一，这与树的定义相悖，假设不成立。

证明3.1.4：L表示一个偶环。

由对称差的结合律可知：$loop_{1+2+\cdots+k} = loop_{((1+2)+3)+\cdots+k)}$。由证明3.1.2：$loop_{1+2+\cdots+k}$是一个由$k$条新增边和部分原树边构成的具有唯一性的环的边集，故$L = loop_{1+2+\cdots+k}$。因为任意两个偶数大小的集合的对称差也是偶数大小，故$loop_{((1+2)+3)+\cdots+k)}$表示偶环。$L$为偶环得证。

由此，在原树中进行若干次操作，每次操作选择两个深度奇偶性不同的节点连边，最终的到的图不包含奇环，故算法3.1.2构造出得图为二分图。具体构造二分图的步骤参见算法 5。

Algorithm 5 Construction of the Bipartite Graph

Input: Number of nodes of the graph n;Number of edge of the graph m
Output: Pairs Arrays edge

```
graph[ ][ ] ← emptyarray
for i ← 2 to n do
    u ← rand(1, i − 1)
    v ← i
    edge append {u, v}
    graph[u][v] = ture
end for
set[2] ← empty array
function DFS(u, father, color)
    top[x] = grand father
    append u to seT[color]
    for j ← 1 to n do
        if graph[u][j] = ture and j ≠ father then
            call DFS(j, u, color x or 1)
        end if
    end for
end function
call DFS(rand(1, n), 0, 0)
for i ← n to m do
    while u ≠ v and {i, j} no exits in edge do
        u = rand(set[0])
        v = rand(set[1])
    end while
    append {u, v} to edge
    if graph[u][j] = ture and j ≠ father then
        call DFS(j, u, color x or 1)
    end if
end for
return edge
```

3.1.2 时空复杂度分析

算法3.1.2空间开销在于存储树边与节点$O(N+M)$，其中N,M是所构造二分图节点与边的数量。时间开销存在分为两部分：搜索整棵树，给每个节点标记深度$O(N)$。从边集中随机选点进行$O(M-N)$次连边,故时间复杂度为$O(M)$。

3.2 仙人掌图的构造

3.2.1 构造仙人掌图的算法及证明

如果一张无向连通图的每条边最多在一个环内，则称它是一棵仙人掌。

设son_u为节点u的子节点编号集合，$len(set)$表示集合set的大小，$father_u$表示节点u的父亲节点编号。若节点P为根，或$len(son_{father\ u})>2$，称P为开始节点。从开始节点出发，至叶子节点结束的路径称为树链。定义有效树链为长度大于 1 的树链。设cnt_u表示以u为根的子树中包含的有效树链的结束节点的最大数量。

算法3.2.1:(基于树链剖分[13]的构造法)：从根节点开始进行向下遍历一棵树。对于节点u，设top_u为节点u所在树链的中深度最浅的节点编号，随机取节点$x\in son_u$。对于$i\in son_u$如果$i=x$，$top_i:=top_u$(" := "表示赋值)，否则$top_i:=i$。若$top_u=top_v$，则u，v属于同一条树链。设set_x表示top值等于x的节点集合。对于每个元素数量大于 2 的集合，随机选择$u,v\in set_x$，添加边$\{u,v\}$至边集。

每条树链中最多只有一个环，最终生成的无向连通图中的每条边最多在一个环内。算法3.2.1可以构造一个高随机性的仙人掌图，但缺点是能构造出的仙人掌图边的最大数量的平均值仅有节点数的1.2倍(见实验3.2.1)。然而上述算法构造的仙人掌图存在密度较低，并且不具有随机性的特点，本文针对上述算法 3.2.1 的缺陷提出了密度高且具备随机性的改进算法及其不劣性证明。

算法3.2.2：从根节点开始进行向下遍历一棵树。对于节点u，设top_u为节点u所在树链的中深度最浅的节点编号，设$\boldsymbol{size_i}$等于以节点$\boldsymbol{i}$为根节点的子树大小。选择节点$\boldsymbol{x}$，$\boldsymbol{x\in son_u}$，$\boldsymbol{size_x=min(size_{son_u})}$。对于$i\in son_u$如果$i=x$，$top_i:=top_u$，否则$top_i:=i$。若$top_u=top_v$，则$u$，$v$属于同一条树链。设$set_x$表示$top$值等于$x$的节点集合。对于每个元素数量大于 2 的集合，随机选择$u,v\in set_x$，添加边$\{u,v\}$至边集。

证明3.2.1：如果$size_u = 1$并且$top_u \neq top_{fathe\ u}$，那么$cnt_u = 0$。如果$size_u = 1$并且$top_u \neq top_{fathe\ u}$，那么$cnt_u = 1$。

如果$top_u \neq top_{fathe\ u}$，$u$为树链开始节点，因为$size_u = 1$，$u$为叶子节点，故该树链长度为 1 不是有效树链，$cnt_u = 0$。如果$top_u = top_{fathe\ u}$，$u$所在的树链至少包含节点$u$与$father_u$，长度大于 1 为有效树链，$cnt_u = 1$。

证明3.2.2：如果$size_u > 1$，无论top_u是否等于$top_{fathe\ u}$，cnt_u不变。

如果$top_u = top_{fathe\ u}$，假设存在一个最优剖分方案$plan$，有$size(plan) = cnt_u$。其中$plan_x$表示以x作为结束节点的树链所包含的点集。设$u \in plan_x$，因为子树大小大于 1，节点u不是叶子节点，故$u \neq x$。因为$top_{fathe\ u} \in plan_x\ \ and\ \ x \in plan_x\ \ and\ \ u \in plan_x$，所以$size(plan_x) > 2$，因此$plan_x$表示一条有效树链。如果$top_u \neq top_{fathe\ u}$，因为$top_{fathe\ u} \notin plan_x$，所以$size(plan_x) > 1$，$plan_x$依然表示一条有效树链，而其他划分方案不变，故$cnt_u$不变。

证明3.2.3：如果存在一个子节点u，$size_u = 1$，那么令总是不劣。

反证法，如果存在一个最优方案，其中存在两个节点u，v，满足$size_u = 1$，$size_v > 1$，$top_v = top_{fathe\ v}$，$top_u \neq top_{father_u}$。设$cnt_v = k$，根据证明3.3.1可知$cnt_u = 0$。如果改变方案令$top_u = top_{fathe\ u}$，$top_v \notin top_{fathe\ v}$，根据证明3.3.1，3.3.2可知此时$cnt_u = 1$，$cnt_v = k$，优于原方案，这与原方案是最优方案的说法相悖。

证明3.2.4：算法3.3.2所剖分出的树链数量不少于算法3.3.1所剖分出的树链数量。

如果在一次剖分中，仅有大小为 1 的子树或者仅有大小大于 1 的子树，根据证明3.2.1与证明3.2.2，无论选择哪个子树根节点u，令$top_u = top_{father_u}$，最终树链数量不变。如果在一次剖分中同时包含两类子树，根据证明3.2.3选择大小只有 1 的子树进行剖分总是更优。综合以上两种情况，选择较小的子树根节点u，令$top_u = top_{father_u}$的方案总是不劣，构造仙人掌图的五算法如算法 6 所示：

Algorithm 6 Construction of the Cactus

Input: Number of nodes of the graph n
Output: Pairs Arrays edge

```
top[ ] = 0
fa[ ] = 0
graph[ ][ ] = 0
for i ← 2 to n do
    u ← rand(1, i − 1)
    v ← i
    edge append {u, v}
    graph[u][v] = ture
end for
function DFS((x, father, grand father))
    top[x] = grand father
    fa[x] = father
    link son ← j where j is any one of graph[x][j] = true
    call DFS(link son, x, grand father)
    for j ← 1 to n do
        if graph[x][j] = true and j ≠ link son then
            call DFS(j, x, j)
        end if
    end for
end function
root ← rand(1, n)
call DFS(root, 0, root)
vis[ ] = false
for i ← 1 to n do
    if vis[top[i]] = false and top[i] ≠ i and top[i] ≠ fa[i] then
        vis[top[i]] = true
        append {i, top[i]}to edge
    end if
end for
return edge
```

3.2.2 时空复杂度分析

复杂度分析：算法3.2.2需要存储所有边$O(M)$以及若干用于记录信息的辅助数组$O(N)$，空间复杂度为$O(N+M)$。构造树，进行树剖与构造环的时间复杂度均为$O(N)$，故总时间复杂度为$O(N)$。

3.2.3 实验结果分析

实验3.2.1：下面用实验对算法3.2.2的优化效果进行对比验证。输入数据为一棵随机构造的树，输出数据为该算法在树上剖分的树链数量。实验数据一共有 7 组，每组有 100 个随机生成的树。分别测试在节点数为 100，500，1000，5000，10000，50000，100000 的情况下，两类算法剖分出的平均树链数量。

表 2 不同节点数下，两类算法剖分出的平均树链数

节点数量	100	500	1000	5000	10000	50000	100000
算法 3.2.1	20.28	107.9	217.49	1088.97	2184.26	10912	21821.5
算法 3.2.2	35.72	182.65	367.03	1835.17	3677.19	18397.4	36788.1
比值	**1.514792**	**1.692771**	**1.687572**	**1.685235**	**1.683495**	**1.685979**	**1.685865**

表格2展示了算法3.2.1与算法3.2.2对同一组数据剖分出的树链数量对比，对于每个节点数量级均准备了 100 组测试数据，测试数据使用算法 2 随机构造。可以看到在不同节点数量级下，算法3.3.2构造出的平均树链数量均较算法3.3.1提高了70%。

图2展示了在不同节点数量下，算法 3.3.1 和算法 3.3.2 进行 100 次随机构造与剖分后得到的随机树链数量结果的折线图。不难看出在不同节点数的情况下，算法 3.3.1 所剖分出的数链数量大致是节点总数的 20%，算法 3.2.2 所剖分出的数链数量大致是节点总数的 35%。如果需要构造更稠密的仙人掌图，则建议使用非随机性构造算法，即由每个节点向其父节点连边构造。相关代码与数据已被开源整理到 https://github.com/gcwAndHisFriends/cactus_construct_check。

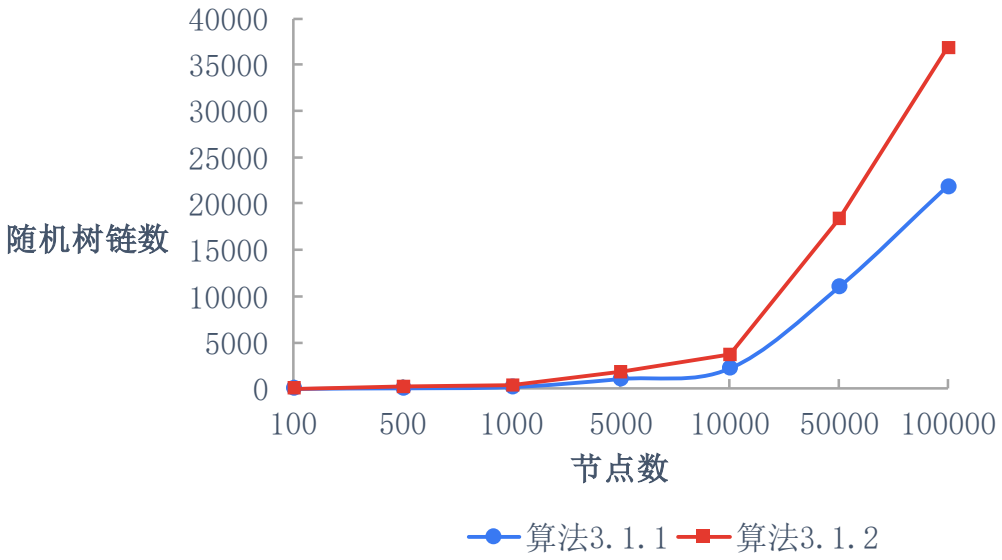


图 2 不同节点数量下，算法 3.3.1 和算法 3.3.2 得到的随机树链数量

4 结论

图数据是未来数据存在形式的新范式。许多的研究基于图数据之上进行复杂的建模和表征学习。针对目前对于图数据的构造算法鲜有研究的问题和先进工作中的图数据构造方法存在一些随机性不强，无法构造边界数据或构造边界数据时时间复杂度会退化的问题。本文提出了兼顾随机性与时间效率的图数据构造算法。具体而言，对树、有向无环图、有向带环图、二分图、仙人掌图等常见的图数据模型，提出了数据范围自适应的算法 1.3、稠密随机仙人掌构造算法 3.2.2，并完成了算法 1.3、算法 3.1.2 与算法 3.2.2 的严密的证明工作，还进一步将算法 1.3 衍生到有向图的构造中，并给出了时空复杂度分析和 C++代码实现。构造图数据的算法在科研、工程开发中可以起到测试算法的正确性和运行效率的作用。然而，本文也存在局限性，比如只考虑了较小数据集的构造方法。我们将探索应用于分布式图数据分析系统的大数据集图数据构造算法作为未来的研究工作。

5 参考文献

[1] Liu H, Zheng C, Li D, et al. Multi-perspective social recommendation method with graph representation learning[J]. Neurocomputing, 2022, 468: 469-481.

[2] Van Belle R, Van Damme C, Tytgat H, et al. Inductive graph representation learning for fraud detection[J]. Expert Systems with Applications, 2022, 193: 116463.

[3] Zhu X, Li Z, Wang X, et al. Multi-modal knowledge graph construction and application: A survey[J]. IEEE Transactions on Knowledge and Data Engineering, 2022.

[4] Hamilton W, Ying Z, Leskovec J. Inductive representation learning on large graphs[J]. Advances in neural information processing systems, 2017, 30.

[5] Zhao J, Zhao Y, Li J, Yan K, Tian Y. Heterogeneous Relational Complement for Vehicle Re-identification[C]//2021 IEEE/CVF International Conference on Computer Vision (ICCV).

[6] 王映龙,宋泽锋,陈卓.基于图的多关系数据挖掘理论研究与方法[J].计算机科学,2008(05):152-153+157.

[7] 曾滔.改进的 k 度匿名图构造算法[J].计算机系统应用,2022,31(05):157-164.

[8] 王桂平、杨建喜、李韧编著. 图论算法理论、实现及应用(第 2 版)[M], 北京大学出版社，2022 年 1 月.

[9] 王桂平，王衍，任嘉辰. 图论算法理论、实现及应用[M]. 北京大学出版社, 2022.

[10] Knuth D E. The art of computer programming[M]. Pearson Education, 1997.

[11] Hopcroft, J. E., & Tarjan, R. E. (1973). Algorithm 447: Efficient algorithms for graph manipulation. Communications of the ACM, 16(6), 372-378.

[12] Wagner, K . Über eine Eigenschaft der ebenen Komplexe. Math. Ann. 114, 570–590 (1937).

[13]Gabow H N . Path-based depth-first search for strong and biconnected components[J]. Information Processing Letters, 2000, 74(3-4):107-114.

作者简介：
葛昌威（2001—），男，重庆丰都人，学士，研究方向为数据科学与大数据
黄春淦（1998—），男，四川广安人，硕士，研究方向为图神经网络、推荐系统
王桂平（1982—），男，广东肇庆人，硕士，讲师，研究方向为图论算法、大规模图数据分析及处理等
通信作者：王桂平（1979—），男，江西安福人，博士，副教授，硕导，研究方向为图论算法、大规模图数据分析及处理等，E-mail：wgp@cqjtu.edu.cn。

非公开发表部分：

姓名：黄春淦
联系地址：重庆交通大学信息科学与工程学院计算机系
电子邮箱：1053538739@qq.com
联系电话：15310970050